**Report - template**

Assignment 3 - MongoDB

**Group**: 51

**Students**: Henrik Borge, Torbjørn Grande, Sondre Rogde

Note: to make grading easier, everything that is different from exercise 2 will be written in cyan. The screenshots are of course also different, but they are not marked in cyan.

**Introduction**

Briefly explain the task and the problems you have solved. How did you work as a group?

The task was to implement a structure for storing data on activities. Each activity was related to a user and potentially multiple trackpoints. The trackpoints contains a timestamp with the coordinates of the user’s position along with altitude. This was done keeping in mind that we were storing data for an application similar to Strava (more on this under Discussion).

All team members knew each other well from before, so working as a group posed no problems. Being a team of three people really helped in discussing the technical details of how to store the data and what we had to consider. For most problems in part 2, we implemented a simple solution assuming the data was clean and made sure that solution worked. After implementing this simple solution, we expanded on it to deal with edge cases, invalid data and other things that could invalidate the output. This is discussed in greater detail under Results and Discussion.

We used Github for code collaboration and version control. The repository on Github can be found here: <https://github.com/Sondringsen/StoreDistribuerteOvinger>. The repository should be publicly available, but please let us know if there is something wrong with the access. The repository also contains a README.md containing all documentation required for running the code.

**Results**

Add your results from the tasks, both as text and screenshots. Short sentences are sufficient.

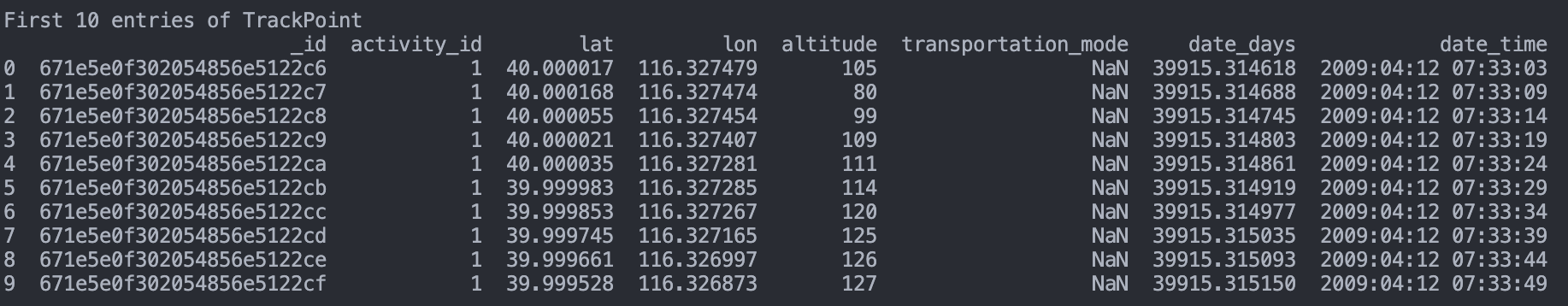
The following is small excerpts from the collections. We decided to implement the database in a slightly different way than the suggested structure to make the application a bit more flexible. This will be further discussed under Discussion.

**A screenshot of a computer screen

Description automatically generated**

**A screen shot of a number

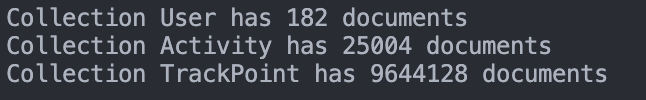
Description automatically generated**

****

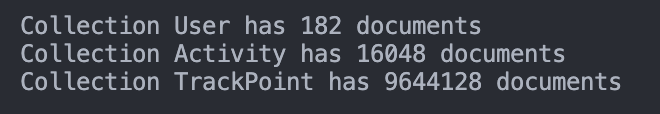
Question 1:

Notice that there is a high number of activities. The reason for this will be discussed in the Discussion and has to do with our goal of making the application similar to Strava. It is clear that when including only activities where trackpoints are registered we get far fewer activities.

Including all activities:

****

Including only activities with trackpoints:



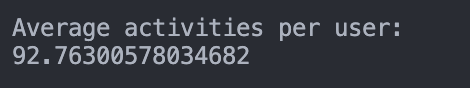
Question 2:

Average activities per user when including all activities:

A white text on a black background

Description automatically generated

Average activities per user when including only activities with trackpoints:



Question 3:

We see that 163 is on top here, but if we only include activities that has trackpoints, 128 would be on top.

Activity count for all activities:

A screenshot of a computer screen

Description automatically generated

Activity count for only activities which have trackpoints:

A screenshot of a computer screen

Description automatically generated

Question 4:

Including all activities:

A screenshot of a computer

Description automatically generated

Only including activities which have trackpoint:

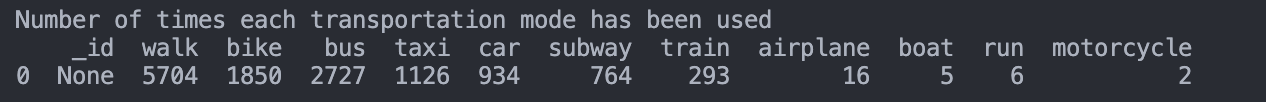
A screenshot of a computer

Description automatically generated

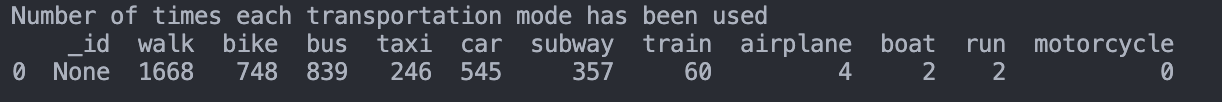
Question 5:

For this question our implementation worked very well since we only had to sum over all the binary variables in the Activity collection.

All activities:



Only including activities with trackpoints:



Question 6a:

In both question 6a and 6b we see that there is data registered for the year 2000. In the description of the dataset, it said that all activities were between 2007-2011, but we have an activity from 2000 and 616 activities from 2012 which does not make sense. However, after taking a closer look at some of these activities, it did not seem like there was anything wrong with them and we decided to keep them. For instance, all the trackpoints to the activity from 2000 is shown below. It is only 3 trackpoints, but they all seem valid. Also, the numbers are quite high here and would be almost halfed if we did not include activities that has no trackpoints.

Total activities:

A screenshot of a computer program

Description automatically generated

Only including activities which have trackpoints:

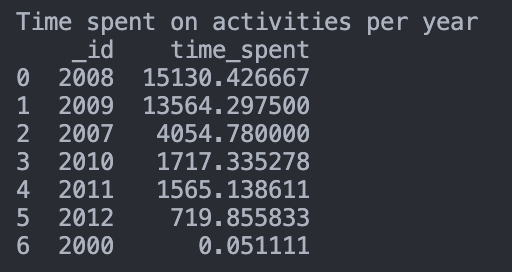
A screenshot of a computer screen

Description automatically generated

Question 6b:

Depending on whether we include only the activities with trackpoints or all the activities the year with most time spent changes from 2008 to 2009.

Time spent on all activities:



Time spent on activities which have trackpoints:

A screenshot of a computer program

Description automatically generated

Question 7:

For this question only 10km moves between two trackpoints were allowed. This was to avoid any faulty data where two consecutive trackpoints was too far from each other. For this specific user, it did not matter, however, it could matter for other users.



Question 8:

For this question only altitudes between -300 and 50,000 feet were allowed. All altitudes of -777 altitudes were dropped. Also, if the altitude changed with more than 300 feet it was discarded. For this question, the only terms in the sum are the terms where there was a positive difference in altitude between two trackpoints, i.e., where the user ascends.

A screenshot of a computer

Description automatically generated

Question 9:

We found that most users have invalid activities.

A screenshot of a computer program

Description automatically generated

Question 10:

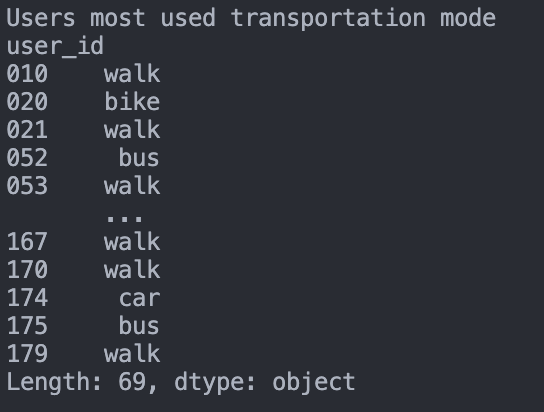
For this question we allowed the latitude to be between 39.915 and 39.917 and the longitude to be between 116.396 and 116.398.

A screen shot of a computer

Description automatically generated

Question 11:

Including all activities:

****

Only including activities with trackpoints:

**A screenshot of a computer program

Description automatically generated**

**Discussion**

Discuss your solutions. Did you do anything differently than how it was explained in the assignment sheet, in that case why and how did that work? Were there any pain points or problems? What did you learn from this assignment?

What we did differently

We did some things a bit differently than what was described in the given description. They were only minor changes implemented to make the solution more realistic for an app similar to Strava. In Strava you can have multiple transportation modes per activity. You can for instance run and cycle in one activity in Strava. To better meet this requirement the table storing activities contained a column for each transportation mode stored as a binary variable. A different solution is to store a comma-separated list, but this is seen as an antipattern in relational databases. In addition, we stored the transportation mode for a given trackpoint as a string. Trackpoints can only have one transportation mode since it is just a point in time and not an interval. This way we still have the ability to keep track of how much a user has walked and allow multiple transportation modes during one activity. We also decided to add a list of activity\_ids to each user to increase performance of a few queries. We decided to not include data about the entire activity as this is a one-to-many relationship and not one-to-a-few. This was complemented by having a user\_id on the many-side, that is each Activity document had a user\_id field. Each TrackPoint had a field with activity\_id, but the Activity documents did not have any reference to TrackPoint. This is because this is a one-to-quintillion relationship. This design scheme is in line with recommendation promoted by MongoDB (Zola, 2022).

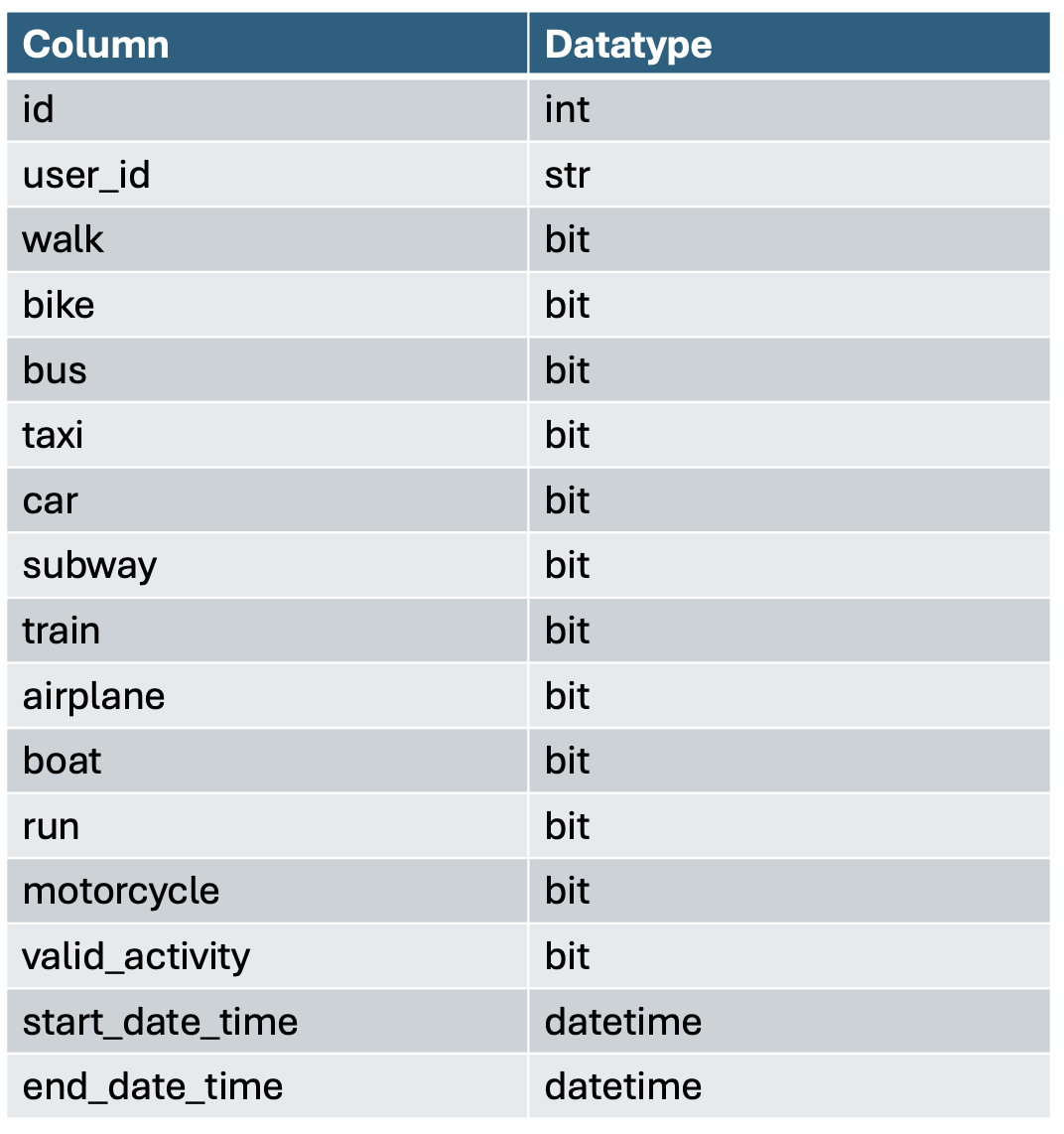
To make the app more similar to Strava where you can register activities without tracking them (with trackpoints), we decided to let an activity be defined in two ways. Either, you have a plt file with trackpoints where all trackpoints is one activity (which can be either labeled or not). Or, you have a labeled activity without any trackpoints. This is also why the number of activities may seem inflated in some questions. If we did not allow this the number of activities would be much lower. In fact, it would only be 16048 as shown below.

INSERT SS

Please see the figures below for a detailed description of the tables and datatypes.

A blue and white rectangular object with black text

Description automatically generated with medium confidence



A table with text on it

Description automatically generated

Handling messy data

To make the functionality most similar to Strava we let any activity be defined by the .plt files with start time and end time equal to the min and max timestamp of that plt.file respectively. If an activity was labeled, but did not correspond to any TrackPoints, we added it as an activity, but flagged it as invalid. We did this because you can add activities in Strava without having any GPS tracking (i.e. no trackpoints). Before inserting the data, any trackpoints with highly unlikely values were removed. For instance, the altitude was restricted to be between -300 and 50,000 feet. The coordinates were also verified to be between -90 and 90 and -180 and 180 for latitude and longitude respectively. There is also some consecutive trackpoints which does not make sense, for instance, where the distance between them is unrealistically high. This is not handled when inserting the data but is handled during the queries where the distance between trackpoint-coordinates must be within a certain range. The same goes for altitude.

Pain points

Most of the exercise was completed without any great obstacles. The only pain point experienced was during part 2 with some of the more complicated queries. Joins are quite easy with relational databases, but not so much with document-based ones. We solved this by retrieving documents and putting them into pandas dataframes which you can easily join. We also noticed that the performance of some of the queries which needed to join tables were better using MongoDB. However, this might be because when we used MongoDB the join was performed on client-side while the joins when using mysql were performed on server-side.

What we learned

We learned a lot doing this project especially since none of the group members had any prior experience with document-based databases. For this specific exercise the schema design was very similar to that of the relational database design. However, it is easy to imagine other use cases where this would not be true, and the design of the document-based design would look very different from a relational-based design. After discussing and thinking about this in the group we think we are better able to determine a good design for a database and decide whether to use a document-based or a relational database.

**Feedback**

Optional - give us feedback on the task if you have any. The assignment is new this semester and we would love to improve if there were any problems.

We really appreciated how this exercise extended exercise 2. It was interesting to see the differences between relational databases and document-based databases in a project like this. It makes comparison easier and makes you think more critically about what database type would be appropriate for other use cases.

**References**

Zola, W. (November 2, 2022). *6 Rules of Thumb for MongoDB Schema Design*. MongoDB. Retrieved: 28.10.2024 from: <https://www.mongodb.com/blog/post/6-rules-of-thumb-for-mongodb-schema-design>